Estfeed Protocol

Protocol Specification

Version: 1.15

15.03.2018

32 pages

Y-1029-1

Date	No	Description	Author
18.02.2015	0.1	Initial draft	Siim Annuk
03.02.2015	0.2	Some minor adjustments	Siim Annuk
04.02.2015	0.3	Reorganized the structure, added messaging	Siim Annuk
06.03.2015	0.4	Added generalized message flow diagram	Siim Annuk
06.03.2015	0.5	Added requirements for applications and sources	Siim Annuk
09.03.2015	0.6	Reorganized text, added examples	Siim Annuk
10.03.2015	0.7	Added sequence diagrams	Siim Annuk
10.03.2015	0.8	Fixed typos and examples	Siim Annuk
11.03.2015	0.9	Corrections regarding message format	Siim Annuk
12.03.2015	1.0	Fixed typos in text and in schema	Siim Annuk
06.04.2015	1.1	Renamed message to request in schema	Siim Annuk
08.06.2015	1.2	Added message schema changes	Siim Annuk
09.06.2015	1.3	Specified processing rules for private data	Siim Annuk
29.04.2016	1.4	Added missing example messages	Siim Annuk
20.05.2016	1.5	Added clarification about HTTP usage	Siim Annuk
27.05.2016	1.6	Clearly mark HTTP header, avoid “subscribe”	Urmet Jänes
01.06.2016	1.6.1	Minor improvements and fixed some typos	Siim Annuk
10.08.2016	1.7	Error messages, error flow descriptions	Urmet Jänes
02.09.2016	1.8	SourceId is optional for data source	Urmet Jänes
16.09.2016	1.9	Estfeed logo added. Mandate objects in all PvtData messages, more generic MO types used	Urmet Jänes
06.12.2016	1.10	Error message from data source may generate an error	Urmet Jänes
22.02.2017	1.11	Response without payloads explained	Urmet Jänes
21.04.2017	1.12	Explained mutual URLs, illustrated TLS and key exchange	Urmet Jänes
12.05.2017	1.13	Added paragraph about calculating message part digests	Siim Annuk

18.08.2017	1.14	Clarified message part digest calculation	Siim Annuk
15.03.2018	1.15	Added application adapter pull method description. Restructured to better explain protocol layers.	Siim Annuk, Urmet Jänes

Table of Contents

1 Introduction

1.1 Purpose

This document describes the protocol and format of the messages that are used to communicate with the Estfeed system. The document also provides example messages and schemas of messages.

1.2 References

1. MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies. Request for Comments: 1341, Internet Engineering Task Force, June 1992.

2 Protocol Overview

2.1 Parties

Data exchange involves 3 parties (Figure 1):

- Application information system (application IS) – consumer of data and services; communicates with the Estfeed system (application adapter) using Estfeed protocol;
- Source information system (data source IS) – provider of data and services; communicates with the Estfeed system (source adapter) using Estfeed protocol;
- Estfeed system – mediator of data and services between applications and data sources.

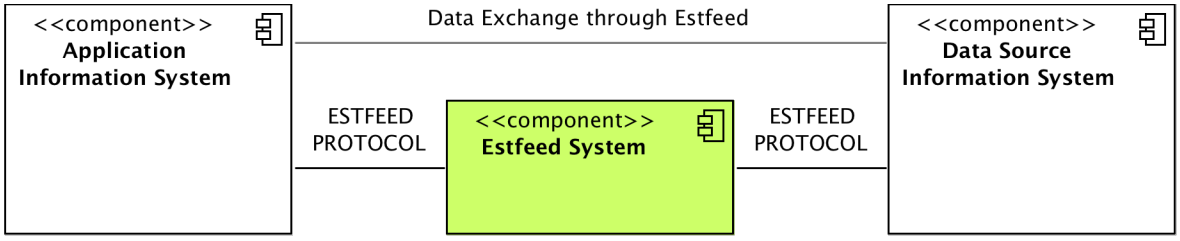

Figure 1. Protocol overview.

The end-to-end data exchange (2.3) uses multipart MIME [1] messages, which consist of an Estfeed protocol message (3) and optional data payloads. Between individual systems, HTTP is used for transfer of the messages (2.2).

2.2 Protocol between Information System and Estfeed

Application information system and source information system exchange data with the Estfeed system over mutually authenticated TLS (HTTPS) connections.

Certificates are exchanged between interconnected systems to achieve the mutual authentication (certificate pinning). In deployment cases, where the information system is deployed to the same host together with the Estfeed system endpoint (Estfeed adapter), regular HTTP can be used for data exchange.

The standard HTTP method is HTTP POST in both directions, where both Estfeed adapter and the information system call an URL to push data (2.2.1).

For an application, an optional pull method is also supported. In this case, the application will periodically poll the Estfeed adapter for incoming messages (2.2.2).

2.2.1 Push Method

The push method of message delivery is the common method applicable to both application and data source Information Systems. The method provides immediate message delivery and low resource utilisation.

Using the push method, each party acts as an HTTP client as well as an HTTP server –

messages to IS are pushed (HTTP POST) to a preconfigured URL (the information system URL is depicted as https://is/url on Figure 2) and outgoing messages to Estfeed are pushed to to the adapter URL

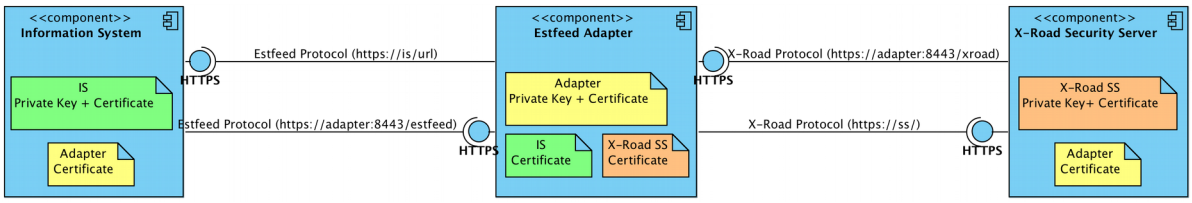

Figure 2. TLS configuration and typical URLs (push method case).

2.2.2 Pull Method for Application Information System

Pull method can only be used by the application information system to pull messages from the Estfeed system, as an alternative to the push method. It is a preferred solution when the application information system cannot accept incoming connections, for instance is behind a firewall . Pull method for application can be enabled in the Estfeed administration application.

With pull method, the application adapter will cache all incoming messages sent from the data sources (either as a response to a request or a publish) in a queue for a time period (in order of minutes) and will not forward (push) the messages to the information system. Instead, the application information system must regularly poll for new messages from the application adapter using HTTP GET request. If a message is not pulled from the adapter message queue before the time period configured in the adapter, the message will expire – it will be removed from the queue and is lost permanently.

The information system uses the HTTP GET request (https://adapter:8443/estfeed) to receive the oldest message stored in the adapter queue. The message is removed from the queue (and from the internal cache in adapter) and sent to the information system. The HTTP GET response contains an HTTP header “Estfeed-Queue-Size” with the number of messages still in the adapter message queue. If the message queue is empty, the HTTP GET request returns an HTTP 204 (no content) status and an empty response body.

When using the pull method, the information system does not send an acknowledgement (3.3) of reception of the message to the adapter. Instead, the adapter treats the pulling of a message from the queue as a successful delivery of the message to the information system.

The application IS should pull messages continuously until the “Estfeed-Queue-Size” value reaches zero, at which point the application IS should start pulling at a sufficiently frequent interval (about 60 seconds) so that the adapter would not run out of memory and disk space when storing messages int the queue under high load.

The pull algorithm for the application IS is as follows :

1. make HTTP GET request;

   a) if the HTTP response status is 204 or other error occured: wait/sleep for a specified amount of time and go to 1;

   b) if the value of response HTTP header “Estfeed-Queue-Size” is greater than

zero: parse HTTP response content as an Estfeed message and go to 1 without waiting.

c) otherwise: parse HTTP response content as an Estfeed message, wait/sleep for a specified amount of time and go to 1;

2.3 Data Exchange Protocol between Information Systems

This section describes the end-to-end protocol interactions while exchanging messages through Estfeed.

2.3.1 Publish Protocol

In publish data exchange, the data sources periodically publish messages (e.g. measurement data) to the Estfeed system. Applications are subscribed to the services publishing the data (note: the subscription process is not part of the Estfeed protocol). When a data source publishes data, the Estfeed system automatically forwards it to all applications that are subscribed to the service provided by the data source and are permitted to receive the data. The applications receive data either via push method (2.2.1) or pull method (2.2.2).

The publish protocol can only be used to receive data that is produced after subscription. For accessing historical data, the request-response protocol (2.3.2) must be used.

The sequence diagram in Figure 3 shows a scenario where a data source publishes data to subscribed application 1, after which application 2 subscription is enabled to the service provided by the source. Finally, the data source publishes data to subscribed applications 1 and 2.

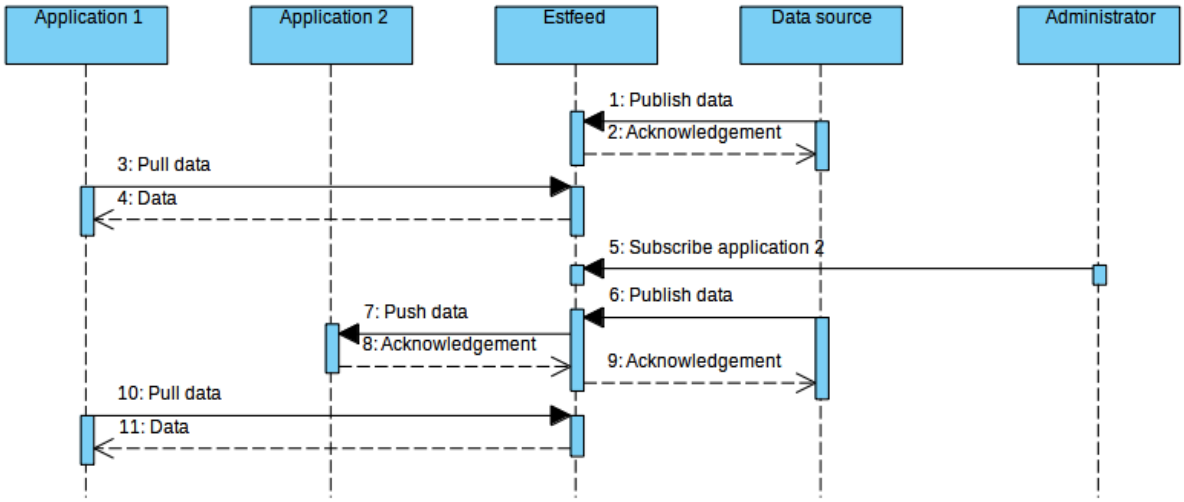

Figure 3. Publish (push and pull method): 2 applications, 1 source.

2.3.2 Request-Response Protocol

The request-response protocol can be used to invoke services offered by data sources. One service can be provided by multiple data sources. Estfeed automatically sends the request to all involved data sources.

Request is initiated by an application that sends a request message to the Estfeed system. The Estfeed system immediately responds with an acknowledgement message containing a transaction identifier and a list of responder source identifiers. The Estfeed system then

forwards the request to data sources that provide the specified service.

Each data source produces the corresponding data message with the transaction identifier previously generated by the Estfeed system. The data message is pushed to the application or made available for pulling.

Data messages without a payload should be used for cases where the service does not produce data, either by nature of the service (successful completion of a command) or when the request is processed without producing an error, but there is no data to return (e.g. no data is available for requested time period).

The sequence diagram in Figure 4 shows how an application sends a data request to the Estfeed system and the Estfeed system forwards the request to data sources 1 and 2. The sources then publish the requested data to the Estfeed system which forwards the data to the application.

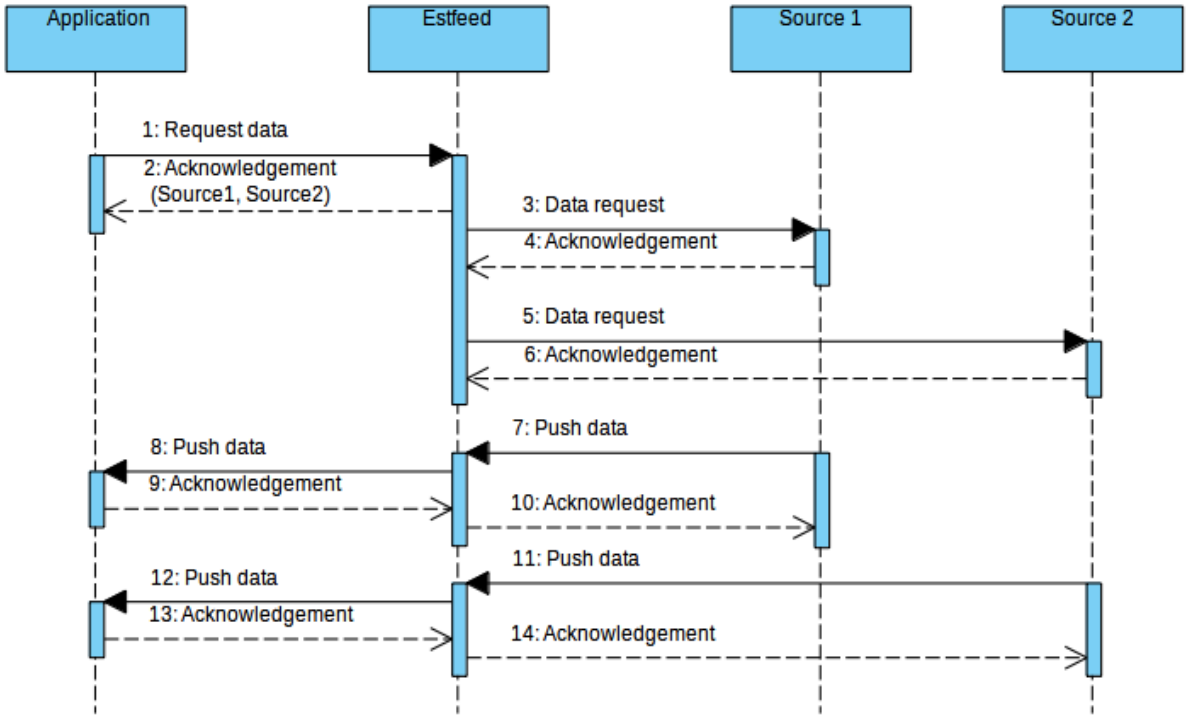

Figure 4. Request-response (push method): 1 application, 2 sources.

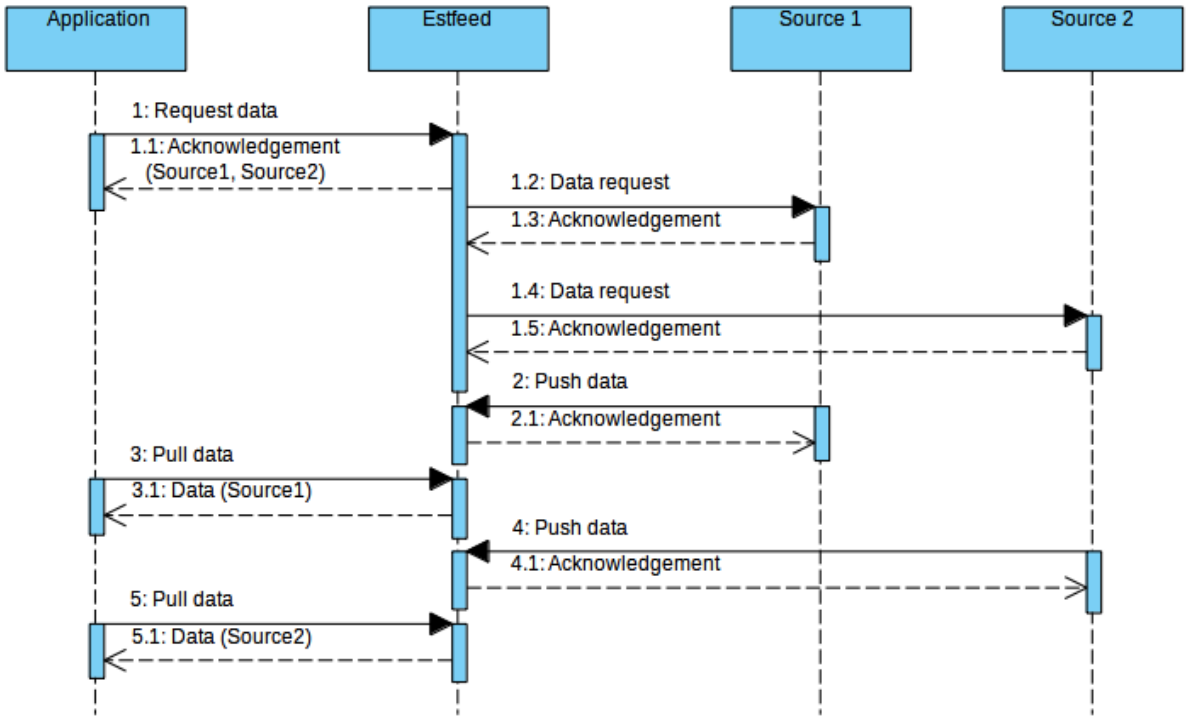

Figure 5. Request-response (pull method): 1 application, 2 sources.

2.4 Error Handling

Each party (application, data source, Estfeed) can reply to an incoming message with an error message, indicating that the message will not be processed further.

An error message may be synchronous (replacing the acknowledgment message) or asynchronous (replacing the data message). The message content provides the error context (related transaction and error source). Any party producing an error message must not include any private data or internal stack traces for security reasons. If there is a need to correlate with system logs of a component, it is suggested to use transaction codes or specially generated identifiers.

An application must also be ready for not receiving a data message (an expected publish or a response) and take corresponding action after a request timeout, e.g raising an error condition or retrying a request. The response time is different for each service and Estfeed does not make any guarantees of response time.

Specific conditions that generate error messages for an **application** are as follows.

1 Protocol	2 Synchronosity	3 Condition
Request	Synchronous	Request invalid
Request	Asynchronous	Estfeed internal processing failure
Request	Asynchronous	Data source processing failure

A **data source** may receive an error message when sending a data message (publish or response) or error message (as a response to a request) that does not pass the Estfeed validation checks. Applications don't receive any indication of these errors.

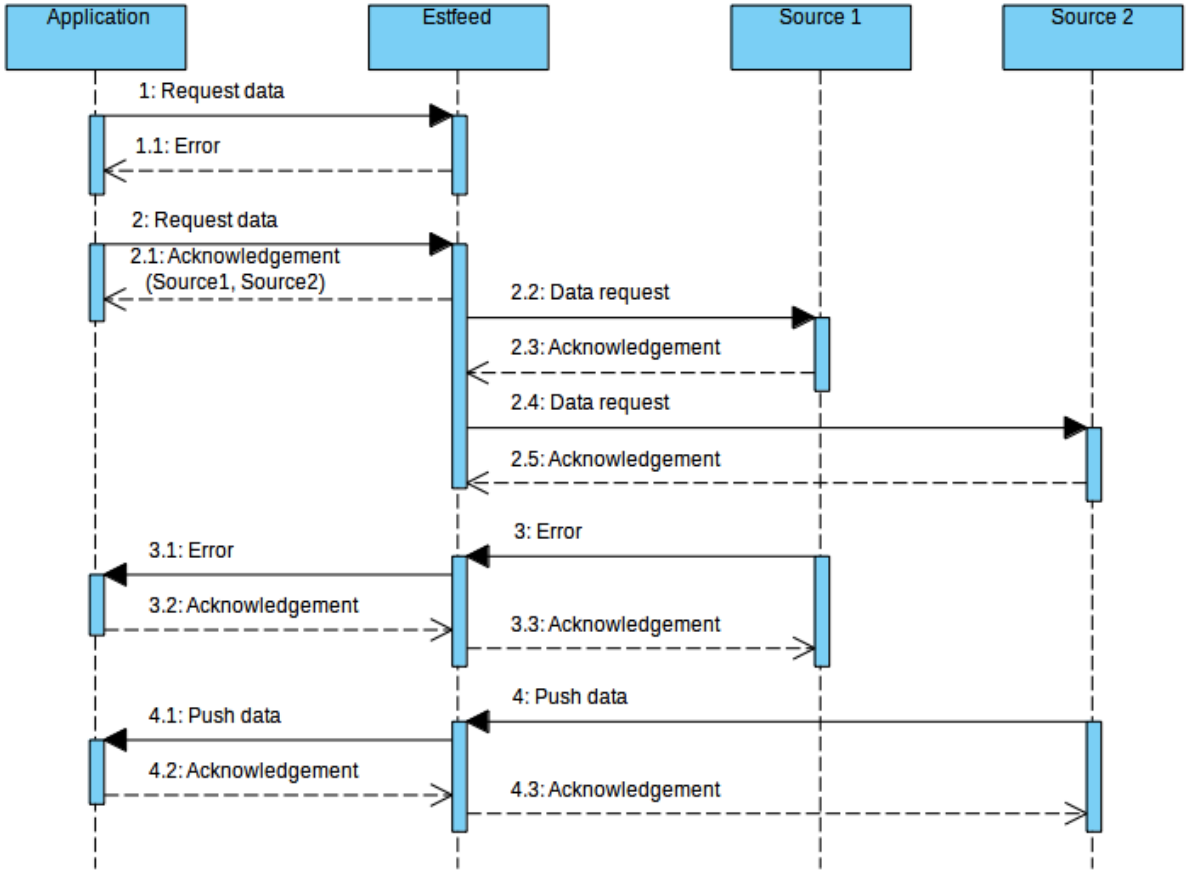

Figure 6. Error message received from Estfeed (Request 1) and from data source (Request 2)

3 Estfeed Message Format

Estfeed system uses four kinds of messages. Request messages (section 3.2), acknowledgement messages (section 3.3), data messages (section 3.4) and error messages (section 3.5).

3.1 Overview

Estfeed system uses HTTP 1.1. protocol. POST request method is used for all requests when using the push method. When using the pull method in the application information system, GET request method must be used.

All Estfeed messages are encoded using MIME multipart (MIME) format. The first part of the multipart message must contain message metadata in XML format. The following parts of the message are service-specific payloads.

The HTTP Content-Type of the message must be „multipart/related”. The body of the message must be in the MIME multipart format.

Multipart message has the following format (where the `boundary` value is a generated string).

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_E03hgAd2inthzQF;
```

HTTP body:

```
--MIME_E03hgAd2inthzQF
Content-Type: text/xml; charset=UTF-8

<...metadata...>
--MIME_E03hgAd2inthzQF

Content-Type: ...

<...payload...>
--MIME_E03hgAd2inthzQF--
```

The content type of the payload part is not specified. Messages without payloads are also valid messages.

3.2 Request Message Metadata Format

Request message metadata has the following format:

```
<estfeed:request xmlns:estfeed="http://estfeed.ee/xsd/estfeed-1.0.xsd">
    <transactionId>...</transactionId>
    <service>
        <code>...</code>
        <version>...</version>
        <kind>...</kind>
    </service>
</estfeed:request>
```

Name	Type	Use	Description
transactionId	string	optional	The unique identifier of the messaging transaction. The value can be used to bring together a request and response. TransactionId is generated automatically in the Estfeed system.
service/code	string	required	The unique identifier of the service.
service/version	string	required	The version of the service.
service/kind	string	required	The type of the service.

3.3 Acknowledgement Message Metadata Format

Acknowledgement message metadata has the following format:

```
<estfeed:acknowledgement xmlns:estfeed="http://estfeed.ee/xsd/estfeed-1.0.xsd">
    <transactionId>...</transactionId>
    <service>
        <code>...</code>
        <version>...</version>
        <kind>...</kind>
    </service>
    <responders>
        <sourceId>...</sourceId>
    </responders>
</estfeed:acknowledgement>
```

Name	Type	Use	Description
transactionId	string	optional	The unique identifier of the messaging transaction. The value can be used to bring together a request and response. TransactionId is generated automatically in the Estfeed system when receiving a request or a publish.
service/code	string	required	The identifier of the service.
service/version	string	required	The version of the service.
service/kind	string	required	The type of the service.
responders/sourceId	string	optional	The identifier of the data source that will respond with data.

3.4 Data Message Metadata Format

Data message metadata has the following format:

```
<estfeed:data xmlns:estfeed="http://estfeed.ee/xsd/estfeed-1.0.xsd">
    <transactionId>...</transactionId>
    <service>
        <code>...</code>
        <version>...</version>
```

```
        <kind>...</kind>
    </service>
    <sourceId>...</sourceId>
</estfeed:data>
```

Name	Type	Use	Description
transactionId	string	optional	The unique identifier of the messaging transaction. A data source must supply the transactionID of the request when sending the response. For a publish, Estfeed always generates a unique transactionID.
service/code	string	required	The identifier of the service.
service/version	string	required	The version of the service.
service/kind	string	required	The type of the service.
sourceId	string	optional	The identifier of the data source. Data sources must not include the sourceId element in the data message. The sourceId element is automatically added to the message by the Estfeed system.

3.5 Error Message Metadata Format

Error message metadata has the following format:

```
<estfeed:error xmlns:estfeed="http://estfeed.ee/xsd/estfeed-1.0.xsd">
    <transactionId>...</transactionId>
    <message>...</message>
    <detail>...</detail>
</estfeed:error>
```

Name	Type	Use	Description
transactionId	string	optional	The unique identifier of the messaging transaction this error message is a part of.
message	string	required	The human readable description of the error.
detail	string	optional	The details of the error.

3.6 Payload Format

Payload parts of the Estfeed message carry service-specific business content.

Payload in request messages carries service call parameters, data message payload carries service response data. Messages without payloads are valid and encouraged for request-response services and use cases where there is no data to transmit in either direction (see also 2.3.2).

Payload parts may contain information system specific MIME headers, such as content type, content transfer encoding etc.

The payload parts containing private data must be formatted as described in section 3.6.1.

3.6.1 Private Data

Payload parts carrying private data must include the following MIME headers in each payload part to identify the mandate object associated with the contained data.

Name	Type	Description
Estfeed-MandateObjectCode	string	The code of the mandate object associated with the payload data.
Estfeed-MandateObjectKind	string	The type of the mandate object associated with the payload data.

Each payload part must contain only the data specific to the mandate object identified in the part header.

When receiving messages containing private data, it is strongly recommended to verify, that the mandate object identifier (code and kind) in the MIME header of the payload part matches the mandate object identifier in the binary data of that payload part.

3.6.2 Open Data

No additional requirements are applied to payload parts carrying open data.

3.7 Message Part Digest Calculation

3.7.1 Overview

Estfeed adapter sends an Estfeed message to X-Road as a single X-Road message where each Estfeed message part becomes an X-Road message attachment. X-Road stores the SHA-512 digests of message attachments in the X-Road message log. This provides the means of proving that a specific message part (payload) was sent over X-Road protocol.

This chapter explains how to calculate digests of Estfeed message parts.

3.7.2 Digest Calculation of a Message Part

Estfeed message is encoded using MIME multipart (MIME) format. The MIME message part consists of MIME headers (i.e. Content-Type) and content. The headers and content are separated by first occurrence of a newline. Any other characters occurring after the newline are considered part of the content.

For example, a MIME message with a single message part:

```
--MIME_boundary
Content-Type: text/plain

Content of the message part.
--MIME_boundary--
```

The SHA-512 digest of the message part must be calculated from the byte contents of the message part. The headers of the message part must not be included in the digest calculation.

For example, the following Estfeed data message highlights the contents from which the

digests are calculated:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8
```

```
<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:data>
```

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Estfeed-MandateObjectCode: foo
Estfeed-MandateObjectKind: UsagePoint
```

```
<... XML data ...>
```

```
--MIME_boundary
Content-Type: text/plain; charset=UTF-8
Estfeed-MandateObjectCode: bar
Estfeed-MandateObjectKind: UsagePoint
```

```


This is a
multi line

contents of a message
part.


```

```
--MIME_boundary--
```

The HEX-encoded SHA-512 digests of the message parts are as follows:

```
46ebadf873066562ca1c51801117747a2ca7f4d7e87c18347ab8bf8a8f289909ce4c5725c1f21dfc23390bf21a630
7699c5e84dfd12a2fa43099c1a2f72a85cf

3aa0d0e4ad7555ae57068962cd09d0f0fd3ee804352fcb7121e4005a51ba29f3af987cac69e4b8f79ae21db87102d
9f2802f280af381a9cb606144dd09639e2d

da0b90481cc2a4e2e308ad4794bce9ecd99e6c7ac858a956d329fc2038ec89c01c0e794a1813aa3f604cb8b81983e
9c19535786461b60376d6cadea315036c46
```

It is important to note that the newlines in the message part contents are part of the digest calculation. This can be illustrated as follows:

```
--MIME_boundary
Content-Type: text/plain; charset=UTF-8
Estfeed-MandateObjectCode: bar
Estfeed-MandateObjectKind: UsagePoint
```

```
\n
This is a\n
multi line\n
\n
contents of a message\n
part.\n
```

```
--MIME_boundary--
```

More specifically, the byte contents of the message parts above are:

Part #1:

```
3c 65 73 74 66 65 65 64  3a 64 61 74 61 20 78 6d
6c 6e 73 3a 65 73 74 66  65 65 64 3d e2 80 9d 68
74 74 70 3a 2f 2f 65 73  74 66 65 65 64 2e 65 65
2f 78 73 64 2f 65 73 74  66 65 65 64 2d 31 2e 30
2e 78 73 64 e2 80 9d 3e  0a 20 20 20 20 3c 74 72
61 6e 73 61 63 74 69 6f  6e 49 64 3e 31 3c 2f 74
72 61 6e 73 61 63 74 69  6f 6e 49 64 3e 0a 20 20
20 20 3c 73 65 72 76 69  63 65 3e 0a 20 20 20 20
20 20 20 20 3c 63 6f 64  65 3e 67 65 74 4d 65 61
73 75 72 65 6d 65 6e 74  44 61 74 61 3c 2f 63 6f
64 65 3e 0a 20 20 20 20  20 20 20 20 3c 76 65 72
73 69 6f 6e 3e 76 31 3c  2f 76 65 72 73 69 6f 6e
3e 0a 20 20 20 20 20 20  20 20 3c 6b 69 6e 64 3e
6d 65 61 73 75 72 65 6d  65 6e 74 44 61 74 61 3c
2f 6b 69 6e 64 3e 0a 20  20 20 20 3c 2f 73 65 72
76 69 63 65 3e 0a 3c 2f  65 73 74 66 65 65 64 3a
64 61 74 61 3e 0a
```

Part #2:

```
3c 2e 2e 2e 20 58 4d 4c  20 64 61 74 61 20 2e 2e
2e 3e
```

Part #3:

```
0a 54 68 69 73 20 69 73  20 61 0a 6d 75 6c 74 69
20 6c 69 6e 65 0a 0a 63  6f 6e 74 65 6e 74 73 20
6f 66 20 61 20 6d 65 73  73 61 67 65 0a 70 61 72
74 2e 0a
```

4 Examples

4.1 Data Exchange: Request Private Data

Protocol: request-response

Description: application requests private measurement data, two data sources respond

4.1.1 Request (application IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:request xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:request>
--MIME_boundary--
```

4.1.2 Acknowledgement (Estfeed system → application IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
    <responders>
        <sourceId>source1</sourceId>
        <sourceId>source2</sourceId>
    </responders>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.1.3 Request (Estfeed system → source1,2 IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:request xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:request>
--MIME_boundary--
```

4.1.4 Acknowledgement (source1 IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.1.5 Data message (source1 IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:data>

--MIME_boundary
```

```
Content-Type: text/xml; charset=UTF-8
Estfeed-MandateObjectCode: foo
Estfeed-MandateObjectKind: UsagePoint

<... XML data ...>
--MIME_boundary--
```

4.1.6 Data message (Estfeed system → application IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
    <sourceId>source1</sourceId>
</estfeed:data>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Estfeed-MandateObjectCode: foo
Estfeed-MandateObjectKind: UsagePoint

<... XML data ...>
--MIME_boundary--
```

4.1.7 Acknowledgement (application IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.1.8 Acknowledgement (source2 IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.1.9 Data message (source2 IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:data>

--MIME_boundary

Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Estfeed-MandateObjectCode: bar
Estfeed-MandateObjectKind: UsagePoint

<... binary data ...>

--MIME_boundary

Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Estfeed-MandateObjectCode: baz
Estfeed-MandateObjectKind: UsagePoint

<... binary data ...>
--MIME_boundary--
```

4.1.10 Data message (Estfeed system → application IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
    <sourceId>source2</sourceId>
</estfeed:data>

--MIME_boundary

Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Estfeed-MandateObjectCode: bar
Estfeed-MandateObjectKind: UsagePoint

<... binary data ...>
--MIME_boundary

Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Estfeed-MandateObjectCode: baz
Estfeed-MandateObjectKind: UsagePoint

<... binary data ...>
--MIME_boundary--
```

4.1.11 Acknowledgement (application IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>1</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.2 Data Exchange: Request Open Data

Protocol: request-response

Description: application requests open data, one data source responds

4.2.1 Request (application IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:request xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <service>
        <code>getOpenData</code>
        <version>v1</version>
        <kind>weather</kind>
    </service>
</estfeed:request>
--MIME_boundary--
```

4.2.2 Acknowledgement (Estfeed system → application IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>2</transactionId>
    <service>
        <code>getOpenData</code>
        <version>v1</version>
        <kind>weather</kind>
    </service>
    <responders>
        <sourceId>source</sourceId>
    </responders>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.2.3 Request (Estfeed system → source IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:request xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>2</transactionId>
    <service>
        <code>getOpenData</code>
        <version>v1</version>
        <kind>weather</kind>
    </service>
</estfeed:request>
--MIME_boundary--
```

4.2.4 Acknowledgement (source IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>2</transactionId>
    <service>
        <code>getOpenData</code>
        <version>v1</version>
        <kind>weather</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.2.5 Data message (source IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>2</transactionId>
    <service>
        <code>getOpenData</code>
        <version>v1</version>
        <kind>weather</kind>
    </service>
</estfeed:data>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<... XML data ...>
--MIME_boundary--
```

4.2.6 Acknowledgement (Estfeed system → source IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>2</transactionId>
    <service>
        <code>getOpenData</code>
        <version>v1</version>
```

```
        <kind>weather</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.2.7 Data message (Estfeed system → application IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>2</transactionId>
    <service>
        <code>getOpenData</code>
        <version>v1</version>
        <kind>weather</kind>
    </service>
    <sourceId>source</sourceId>
</estfeed:data>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<... XML data ...>
--MIME_boundary--
```

4.2.8 Acknowledgement (source IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>2</transactionId>
    <service>
        <code>getOpenData</code>
        <version>v1</version>
        <kind>weather</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.3 Data Exchange: Request Action on Private Data

Protocol: request-response

Description: application initiates a service call with private parameter data. Service responds with private response data.

4.3.1 Request (application IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:request xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <service>
        <code>clientNotify</code>
        <version>v1</version>
        <kind>action</kind>
    </service>
</estfeed:request>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Estfeed-MandateObjectCode: 1234567890
Estfeed-MandateObjectKind: device

<... XML data ...>
--MIME_boundary--
```

4.3.2 Acknowledgement (Estfeed system → application IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>3</transactionId>
    <service>
        <code>clientNotify</code>
        <version>v1</version>
        <kind>action</kind>
    </service>
    <responders>
        <sourceId>source</sourceId>
    </responders>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.3.3 Request (Estfeed system → source IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:request xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>3</transactionId>
    <service>
        <code>clientNotify</code>
        <version>v1</version>
```

```
        <kind>action</kind>
    </service>
</estfeed:request>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Estfeed-MandateObjectCode: 1234567890
Estfeed-MandateObjectKind: device

<... XML data ...>
--MIME_boundary--
```

4.3.4 Acknowledgement (source IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>3</transactionId>
    <service>
        <code>clientNotify</code>
        <version>v1</version>
        <kind>action</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.3.5 Data message (source IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>3</transactionId>
    <service>
        <code>clientNotify</code>
        <version>v1</version>
        <kind>action</kind>
    </service>
</estfeed:data>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Estfeed-MandateObjectCode: 1234567890
Estfeed-MandateObjectKind: device

<... data source specific message ...>
--MIME_boundary--
```

4.3.6 Acknowledgement (Estfeed system → source IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>3</transactionId>
    <service>
        <code>clientNotify</code>
        <version>v1</version>
        <kind>action</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.3.7 Data message (Estfeed system → application IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>3</transactionId>
    <service>
        <code>clientNotify</code>
        <version>v1</version>
        <kind>action</kind>
    </service>
    <sourceId>source1</sourceId>
</estfeed:data>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Estfeed-MandateObjectCode: 1234567890
Estfeed-MandateObjectKind: device

<... data source specific message ...>
--MIME_boundary--
```

4.3.8 Acknowledgement (application IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>3</transactionId>
```

```
    <service>
        <code>clientNotify</code>
        <version>v1</version>
        <kind>smsSending</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.4 Data Exchange: Publish Private Data

Protocol: publish

Description: example message that is published by a data source and pushed to an application

4.4.1 Data message (source IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:data>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Estfeed-MandateObjectCode: foo
Estfeed-MandateObjectKind: UsagePoint

<... XML data ...>
--MIME_boundary--
```

4.4.2 Acknowledgement (Estfeed system → source IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>4</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

4.4.3 Data message (Estfeed system → application IS)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:data xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>4</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
    <sourceId>source</sourceId>
</estfeed:data>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Estfeed-MandateObjectCode: foo
Estfeed-MandateObjectKind: UsagePoint

<... XML data ...>
--MIME_boundary--
```

4.4.4 Acknowledgement (application IS → Estfeed system)

HTTP header:

```
Content-Type: multipart/related; boundary=MIME_boundary;
```

HTTP body:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<estfeed:acknowledgement xmlns:estfeed=”http://estfeed.ee/xsd/estfeed-1.0.xsd”>
    <transactionId>4</transactionId>
    <service>
        <code>getMeasurementData</code>
        <version>v1</version>
        <kind>measurementData</kind>
    </service>
</estfeed:acknowledgement>
--MIME_boundary--
```

5 Appendix A: Message Format Schema

message.xsd:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://estfeed.ee/xsd/estfeed-1.0.xsd"
    targetNamespace="http://estfeed.ee/xsd/estfeed-1.0.xsd">
    <xs:element name="request">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="transactionId" type="xs:string" minOccurs="0"/>
                <xs:element name="service" type="tns:ServiceId"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="acknowledgement">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="transactionId" type="xs:string" minOccurs="0"/>
                <xs:element name="service" type="tns:ServiceId"/>
                <xs:element name="responders" type="tns:Responders" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="data">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="transactionId" type="xs:string"/>
                <xs:element name="service" type="tns:ServiceId"/>
                <xs:element name="sourceId" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="error">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="transactionId" type="xs:string" minOccurs="0"/>
                <xs:element name="message" type="xs:string"/>
                <xs:element name="detail" type="xs:string" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="Responders">
        <xs:sequence>
            <xs:element name="sourceId" type="xs:string"
                minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
     <xs:complexType name="ServiceId">
        <xs:sequence>
            <xs:element name="code" type="xs:string"/>
            <xs:element name="version" type="xs:string"/>
            <xs:element name="kind" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```